



Article

A Bio-Inspired Model of Picture Array Generating P System with Restricted Insertion Rules

Gexiang Zhang ^{1,*} , G. Samdanielthompson ² , N. Gnanamalar David ^{3,†}, Atulya K. Nagar ³ and K.G. Subramanian ^{3,†}

¹ Research Center for Artificial Intelligence, Chengdu University of Technology, Chengdu 610059, China

² Department of Mathematics, Hindustan College of Arts and Science, Rajiv Gandhi Salai (OMR), Padur, Kelambakkam, Chennai 603103, India; samdanielthompson@gmail.com

³ School of Mathematics, Computer Science and Engineering, Liverpool Hope University, Liverpool L16 9JD, UK; ngdmcc@gmail.com (N.G.D.); atulya.nagar@hope.ac.uk (A.K.N.); kgsmani1948@gmail.com (K.G.S.)

* Correspondence: zhanggexiang19@cdut.edu.cn

† Honorary Visiting Professor.

Received: 17 June 2020; Accepted: 17 November 2020; Published: 23 November 2020



Abstract: In the bio-inspired area of membrane computing, a novel computing model with a generic name of P system was introduced around the year 2000. Among its several variants, string or array language generating P systems involving rewriting rules have been considered. A new picture array model of array generating P system with a restricted type of picture insertion rules and picture array objects in its regions, is introduced here. The generative power of such a system is investigated by comparing with the generative power of certain related picture array grammar models introduced and studied in two-dimensional picture language theory. It is shown that this new model of array P system can generate picture array languages which cannot be generated by many other array grammar models. The theoretical model developed is for handling the application problem of generation of patterns encoded as picture arrays over a finite set of symbols. As an application, certain floor-design patterns are generated using such an array P system.

Keywords: membrane computing; P system; array languages

1. Introduction

In the area of membrane computing [1–3], the novel computing model of P systems, introduced by Păun [4], has served as a framework for dealing with problems in different areas of applications [5], such as optimization problems [6–9], robots [10–12], power systems fault diagnosis [13–15], modelling complex market interactions [16], image process [17] and clustering [18], and for Turing computing power [19–22] and computing efficiency [23–28], both in its basic form and in its several variants. One such area of application of P systems is formal language theory with different models of P systems having been developed [29–32] for handling the problem of generation of classes of languages, starting with the seminal work of Păun [4]. In the extension of language theory to two dimensions, P systems have also played a significant role with different kinds of P systems with array objects and array evolving rules having been introduced (see, for example, [33–35]).

The operation of insertion on words [36–39] has been studied in string language theory in the context of DNA computing [40]. Fujioka [41] considered this operation in two-dimensional picture arrays and introduced a picture array generating model. This model has a feature which is analogous to the pure 2D context-free grammar [42] where the operation is rewriting in parallel all symbols in a column or a row of a (rectangular) picture array by strings of equal length while in [41], insertion (instead of rewriting) of strings of equal length is done in parallel between columns or between rows.

Here we consider a restricted type of insertion rules with the “contexts” of length at most one, in the regions of a cell-like P system with the objects in the regions being rectangular picture arrays. The main objective of this study is to develop a new theoretical model of an array P system based on the bio-inspired computing model in the area of membrane computing, endowed with the capability to handle the problem of generation of two-dimensional patterns encoded as picture arrays. We compare the generative power of the resulting array P system model introduced here with other well-investigated picture array generating models. We also exhibit an application to generation of floor-design patterns, called “kolam” patterns [43]. A preliminary version of this paper was presented in the conference ACMC 2018 [44].

In Section 2, needed notions and notations relevant to the study undertaken are briefly described, with Section 2.1 providing the basic notions and Section 2.2 recalling picture-insertion system in a restricted form. In Section 3, the new model of array P system with restricted picture insertion rules (APRPIS) is introduced. In Section 4, APRPIS with one and two membranes are compared. In Section 5, comparison with Pure 2D context-free grammars is done while in Section 6, comparison with certain standard array generating models is carried out. Section 7 provides an application of the new model to generation of floor-designs. In the final Section 8, concluding remarks are given.

2. Preliminaries

For notions related to formal language theory, the reader can refer to [45] and to [42,46,47] for two-dimensional array grammars and languages. For P systems and array P systems, we refer to [4,34].

2.1. Basic Notions

A finite set V of symbols is called an alphabet. A word or a string $\alpha = a_1a_2 \dots a_m$, $a_i \in V$, $1 \leq i \leq m$, ($m \geq 1$) of length m over an alphabet V is a finite sequence of symbols belonging to V . The length of a word α is denoted by $|\alpha|$. The set of all words over V , including the empty word λ with no symbols, is denoted by V^* . For any word $w = a_1a_2 \dots a_n$, ${}^t w$ is the vertical word with the word w written

b

vertically. For example, if $\alpha = bab$ over the alphabet $\{a, b\}$, then ${}^t \alpha$ is $\begin{smallmatrix} a \\ b \\ a \end{smallmatrix}$. If w is a single symbol of the

b

alphabet, then we write ${}^t w$ as w itself. A rectangular $p \times q$ array (also called picture array) X over an alphabet V is of the form

$$X = \begin{matrix} a_{11} & \cdots & a_{1q} \\ \vdots & \ddots & \vdots \\ a_{p1} & \cdots & a_{pq} \end{matrix}$$

where each $a_{ij} \in V$, $1 \leq i \leq p$, $1 \leq j \leq q$. The number of rows of X and the number of columns of X are respectively denoted by $|X|_r$ and $|X|_c$. The set of all rectangular arrays over V is denoted by V^{**} , which includes the empty array λ . $V^{++} = V^{**} - \{\lambda\}$. A picture language is a subset of V^{**} .

The column catenation $X \circ Y$ of two arrays X and Y with the same number of rows is formed by juxtaposing Y to the right of X . The row catenation $X \diamond Y$ of two arrays X and Y with the same number of columns is formed by juxtaposing Y below X . For example, if

$$X = \begin{matrix} a & b & a & a & b \\ b & b & b & a & a \\ a & a & b & a & b \\ a & a & b & b & b \end{matrix}, Y = \begin{matrix} b & b & a \\ b & b & b \\ a & a & b \\ a & b & b \end{matrix}, Z = \begin{matrix} b & b & a & b & b \\ b & a & b & a & b \\ a & a & a & b & b \end{matrix},$$

then

$$X \circ Y = \begin{array}{cccccccc} & & & & & & & \\ & & & & & & & \\ & & & & & & & \\ a & b & a & a & b & b & b & a \\ b & b & b & a & a & b & b & b \\ a & a & b & a & b & a & a & b \\ a & a & b & b & b & a & b & b \end{array}, X \diamond Z = \begin{array}{ccccc} a & b & a & a & b \\ b & b & b & a & a \\ a & a & b & a & b \\ a & a & b & b & b \\ b & b & a & b & b \\ b & a & b & a & b \\ a & a & a & b & b \end{array}.$$

2.2. Picture-Insertion System

A picture-insertion system has been considered in [41] with insertion in a picture array being done between columns or between rows, extending the notion of insertion in words [40]. We recall the definition of this system [41] with a special case, namely, with “contexts” of length one in the rules and call this system as a restricted picture-insertion system.

Definition 1. A restricted picture-insertion system (RPIS) is a 4-tuple $\Gamma = (\Sigma, I_c, I_r, A)$ where

- (i) Σ is an alphabet;
- (ii) $I_c = \{c_i \mid 1 \leq i \leq m\}$, ($m \geq 1$) where c_i , called a column insertion table, is a finite set of column insertion rules with alphabetic contexts of the form (a, α, b) , $a, b \in \Sigma \cup \{\lambda\}$, $\alpha \in \Sigma^*$ such that for any two rules $(a_1, \alpha, b_1), (a_2, \beta, b_2)$ in c_i , we have $|\alpha| = |\beta|$ and either both the left contexts a_1 and a_2 are in Σ (likewise the right contexts b_1 and b_2 are in Σ) or both are λ ;
- (iii) $I_r = \{r_j \mid 1 \leq j \leq n\}$, ($n \geq 1$) where r_j , called a row insertion table, is a finite set of row insertion rules with alphabetic contexts of the form $(d, {}^t\gamma, e)$, $d, e \in \Sigma \cup \{\lambda\}$, $\gamma \in \Sigma^*$ such that for any two rules $(d_1, {}^t\gamma, e_1), (d_2, {}^t\delta, e_2)$ in r_j , we have $|\gamma| = |\delta|$ and either both the up contexts d_1 and d_2 are in Σ (likewise the down contexts e_1 and e_2 are in Σ) or both are λ ;
- (iv) $A \subseteq \Sigma^{**} - \{\lambda\}$ is a finite set of axiom arrays.

For picture arrays $P, Q \in \Sigma^{**}$, a one-step derivation in the RPIS Γ , denoted by $P \Rightarrow Q$, yields Q from P whenever either (i) or (ii) holds: (i) if ${}^t a_1 \cdots {}^t a_m$ and ${}^t b_1 \cdots {}^t b_m$ are two adjacent columns, namely, columns j and $j + 1$ for some $j, 1 \leq j \leq m - 1$, in the picture array P of size $m \times n$, and if $(a_i, \alpha_i, b_i), 1 \leq i \leq m$ are column insertion rules in a column insertion table in I_c , then the rules can be applied in parallel by inserting α_i in the i th row between a_i and b_i for all $i, 1 \leq i \leq m$ or (ii) if $d_1 \cdots d_n$ and $e_1 \cdots e_n$ are two adjacent rows, namely, rows k and $k + 1$ for some $k, 1 \leq k \leq m - 1$, in the picture array P of size $m \times n$, and if $(d_i, {}^t\beta_i, e_i), 1 \leq i \leq n$ are row insertion rules in a row insertion table in I_r , then the rules can be applied in parallel by inserting ${}^t\beta_i$ in the i th column between e_i and d_i for $1 \leq i \leq n$. Likewise insertions to the immediate left or right (respy. immediate up or down) of a column (respy. row) in the picture array P can be defined by requiring the corresponding left or right or up or down contexts in the rules used, to be λ .

The picture language $L(\Gamma)$ generated by Γ consists of picture arrays derived in one or more finite number of derivation steps starting with an axiom array in A . The family of picture languages generated by RPIS is denoted by RPIL.

Example 1. Consider the RPIS $\Gamma_1 = (\Sigma, I_c, I_r, A)$ where $\Sigma = \{a, b\}$, $I_c = \{c_1\}$, $I_r = \{r_1\}$, where

$$c_1 = \{(a, ab, b)\}, r_1 = \{(a, a, \lambda), (b, b, \lambda)\}$$

and A consists of the array $\begin{array}{cc} a & b \\ a & b \end{array}$.

Γ_1 generates a picture language L_1 consisting of picture arrays $X \circ Y$ with X over a and Y over b and $|X|_r = |Y|_r, |X|_c = |Y|_c$. A member of L_1 is shown in Figure 1.

a	a	a	b	b	b
a	a	a	b	b	b
a	a	a	b	b	b
a	a	a	b	b	b

Figure 1. A picture array in the language L_1 .

Starting with the axiom array in A , in any step of a derivation, the application of the column insertion rule of c_1 inserts ab in every row between a and b while the application of the row insertion rule of r_1 inserts in every column and just below a row, the symbol a below a and the symbol b below b . In other words, for example, a derivation in which rules of c_1 and then rules of r_1 are applied, is shown below:

a	a	b	b	\Rightarrow^{c_1}	a	a	a	b	b	b	\Rightarrow^{r_1}	a	a	a	b	b	b
a	a	b	b		a	a	a	b	b	b		a	a	a	b	b	b
a	a	b	b		a	a	a	b	b	b		a	a	a	b	b	b
												a	a	a	b	b	b

3. Array P System with Restricted Picture Insertion Rules

Linking the two areas, namely, membrane computing and picture languages, an array P system with array-rewriting rules, was introduced in [33], which is one of the earliest models in this topic. This model motivated extensive research with several variants of array P systems being introduced based on different considerations (See, for example, [35] and references therein). We now introduce a new kind of array P system with picture arrays as objects and with restricted picture insertion rules in the regions.

Definition 2. An array P System with restricted picture insertion rules (APRPIS) is $\Pi = (\Sigma, \mu, F_1, \dots, F_m, R_1, \dots, R_m, i_0)$, where: Σ is the alphabet, μ is a membrane structure with m membranes labelled in a one-to-one way with $1, 2, \dots, m$; F_i , $1 \leq i \leq m$, is a finite set (can be empty) of axiom picture arrays over Σ in the i th region of μ ; R_i , $1 \leq i \leq m$ is a finite set of column or row insertion tables as in a RPIS in the i th region of μ ; each of the tables has an attached target here, out, in. (as usual, here is omitted and is understood). i_0 is the label of an elementary membrane of μ (the output membrane).

A computation in APRPIS is defined in the same way as in an array-rewriting P system [34,35] with the successful computations being the halting ones: each array, from each region of Π , which can be obtained by applying the restricted picture insertion rules to the arrays associated with that region, should be obtained but rules of only one table is applied; the array obtained in a region is placed in the region indicated by the target associated with the table of rules used; the target here indicates that the array remains in the same region, out indicates that the array is sent to the immediate outer region except for the outermost skin membrane in which case the array sent out is “lost” in the environment; and in indicates that the array is sent to the immediately inner membrane, nondeterministically chosen (but if no inner membrane exists, then the table of rules with the target indication in cannot be used). A computation is successful only if it stops and a configuration is reached where no table of rules can be applied to the existing arrays. The result of a halting computation consists of the arrays placed in the output membrane with label i_0 in the halting configuration. The set of all such arrays computed or generated by the system Π is denoted by $AL(\Pi)$. The family of all array languages $AL(\Pi)$ generated by systems Π as above, with at most m membranes, is denoted by $AP_m(RPIS)$.

We illustrate with an example.

Example 2. Consider the APRPIS $\Pi_1 = (\Sigma, [{}_1[{}_2]_2]_1, F_1, F_2, R_1, R_2, 2)$ where $\Sigma = \{a, b, e\}$, $F_1 = \left\{ \begin{smallmatrix} a & b \\ a & b \end{smallmatrix} \right\}$, $F_2 = \emptyset$. R_1 consists of the column insertion tables (c_1, in) , (c_2, in) and R_2 consists of row insertion table (r, out) where

$$c_1 = \{(a, ab, b)\}, c_2 = \{(a, e, b)\}, r = \{(a, a, \lambda), (b, b, \lambda)\}.$$

In a computation in Π_1 , since only the region 1 has an axiom picture array $\begin{smallmatrix} a & b \\ a & b \end{smallmatrix}$, application of the rule of the table c_1 will insert ab between a and b in every row and the resulting array is sent to region 2 due to target indication in . Application of the rules in r in region 2 inserts a row below one of the rows with a below a and b below b and the array is sent back to region 1 due to target indication out . The process can repeat. If in region 1, the rule of the table c_2 is applied then e is inserted between a and b in every row resulting in an array of the form

$$\begin{array}{ccccccc} a & \cdots & a & e & b & \cdots & b \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ a & \cdots & a & e & b & \cdots & b \end{array}$$

and the array is sent to the output region 2. The computation halts as no other rule could be applied. Note that in the rules in r there is no rule that allows insertion below e and hence the rules of the table r can no longer be applied. The arrays generated are of the form $M_1 \circ M_2 \circ M_3$ where M_1 is an array over a , M_3 is an array over b and M_2 is an array with only one column of e 's. Also, $|M_1|_c = |M_3|_c = |M_1 \circ M_2 \circ M_3|_r$. In otherwords, the number of rows of the array generated equals the number of columns of M_1 which equals the number of columns of M_3 . One such array is shown in Figure 2.

$$\begin{array}{ccccccc} a & a & a & e & b & b & b \\ a & a & a & e & b & b & b \\ a & a & a & e & b & b & b \end{array}$$

Figure 2. A picture array generated by Π_1 .

4. A Hierarchy between One Membrane and Two Membranes

We now compare the families $AP_1(RPIS)$ and $AP_2(RPIS)$ and establish that $AP_1(RPIS)$ is properly included in $AP_2(RPIS)$. It remains to explore whether the hierarchy collapses or not for higher number of membranes.

Theorem 1.

$$RPIL = AP_1(RPIS) \subset AP_2(RPIS)$$

Proof. In a $RPIS$ as well as $APRPIS$ with one membrane, the rules are only the $RPIS$ rules and so it is clear that $RPIL = AP_1(RPIS)$. Also by definition, the inclusion $AP_1(RPIS) \subseteq AP_2(RPIS)$ follows. The proper inclusion follows from Example 2 on noting that in any picture array M generated, the number of columns = $2 \times$ the number of rows + 1. If there is only one membrane, there is no way to control application of the rules of the column and row tables as all the tables of rules are in the same region. \square

5. Comparison with Pure 2D Context-Free Grammars

We now compare the generative power of $APRPIS$ with certain similar picture generating models. The insertion rules considered both in $RPIS$ and $APRPIS$ are restricted forms of the insertion rules considered in the picture insertion system defining the class $INPA$ [41]. Yet we show that there are

picture array languages generated by *APRPIS* with two membranes that are not in the class *INPA*. Also, in [42], a picture array generating model, called pure 2D context-free grammar (*P2DCFG*), is proposed and the generative power of this model is investigated. In this model, all symbols in a column or in a row of a $m \times n$ picture array are rewritten by pure context-free rules [42] of the form $a \rightarrow \alpha$ with α in all the rules in a table having the same length. The family of languages generated by *P2DCFG* is denoted by *P2DCFL*. Two variants of *P2DCFG*, called $(l/u)P2DCFG$ and $(r/d)P2DCFG$ which are incomparable with *P2DCFG* have also been considered [48,49]. In a $(l/u)P2DCFG$ the symbols in the leftmost column or the uppermost row only are rewritten while in a $(r/d)P2DCFG$ the symbols in the rightmost column or the lowermost row are rewritten. We show that there are picture array languages that can be generated by *APRPIS* with two membranes but cannot be in *P2DCFL*, $(l/u)P2DCFL$ and $(r/d)P2DCFL$.

Theorem 2.

1. $AP_2(RPIS) - P2DCFL \neq \emptyset$
2. $AP_2(RPIS) - (l/u)P2DCFL \neq \emptyset$
3. $AP_2(RPIS) - (r/d)P2DCFL \neq \emptyset$
4. $AP_2(RPIS) - INPA \neq \emptyset$

Proof. Consider the picture array language L_2 consisting of $m \times (3n + 2)$ arrays ($m \geq 2, n \geq 1$), such that an array in L_2 is of the form $M_1 \circ M_2 \circ M_3 \circ M_4 \circ M_5$ where M_1, M_3, M_5 are $m \times n$ arrays over $\{a\}, \{b\}, \{d\}$ respectively and M_2, M_4 are $m \times 1$ arrays over $\{e\}$. The language L_2 cannot belong to any of the families *P2DCFL*, $(l/u)P2DCFL$ and $(r/d)P2DCFL$ because in *P2DCFG* generating in different modes, a language in any of these families, symbols in an array can be rewritten in only one column at a time. The language L_2 cannot also belong to the class *INPA* for a similar reason. This means that the array with columns of a 's in the beginning, columns of b 's in the middle and columns of d 's in the end, equal in number, cannot be generated.

The language L_2 is generated by the *APRPIS* Π_2 with two membranes where

$$\Pi_2 = (\{a, b, d, e\}, [1 \ 2]_2, F_1, F_2, R_1, R_2, 2)$$

with $F_1 = \left\{ \begin{smallmatrix} a & b & d \\ a & b & d \end{smallmatrix} \right\}$, $F_2 = \emptyset$. R_1 consists of the column insertion table (c_1, in) and a row insertion table $(r, here)$ and R_2 consists of the column insertion tables $(c_3, out), (c_2, here), (c_4, here)$ where

$$c_1 = \{(a, ab, b)\}, c_2 = \{(a, e, b)\},$$

$$c_3 = \{(b, d, d)\}, c_4 = \{(b, be, d)\},$$

and

$$r = \{(a, a, \lambda), (b, b, \lambda), (d, d, \lambda)\}.$$

The region 1 alone has an axiom picture array. So a computation in Π_2 can start in this region with the applicable tables of rules. If the rule of column insertion table c_1 is applied then ab will be inserted in every row between two adjacent columns of a 's and b 's and the array is sent to region 2. If in this region the rule of column insertion table c_3 is applied then d will be inserted in every row between two adjacent columns of b 's and d 's and the array is sent back to region 1. The process can repeat. At any time, the rule of the row insertion table r can be applied, thus inserting in the picture array a row of the form $a \cdots ab \cdots bd \cdots d$. If at any time, instead of c_3 , the rule of the column insertion table c_2 (resp. c_4) is applied in region 2, then a column of e 's will be inserted between two adjacent columns of a 's and b 's (resp. columns of b 's and d 's). A correct sequence of application of the rules is to apply now in region 2, the rule of the column table c_2 followed by c_4 or c_4 followed by c_2 , thus generating a picture

array of the form shown in Figure 3. Region 2 is the output region and arrays collected here form the picture array language generated which is L_2 . \square

$$\begin{array}{cccccccccccc} a & \cdots & a & e & b & \cdots & b & e & d & \cdots & d \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ a & \cdots & a & e & b & \cdots & b & e & d & \cdots & d \end{array}$$

Figure 3. A picture array generated by Π_2 .

6. Comparison with Certain Standard 2D Grammar Models

A well-known class of two-dimensional picture array languages, called the family of context-sensitive matrix languages (CSML), was introduced by Siromoney et al. [50]. The corresponding class of grammars, known as CSMG involves two phases of generation with the first phase generating a string s over intermediate symbols and in the second phase all the intermediate symbols in s are rewritten in parallel by regular nonterminal rules of the form $A \rightarrow aB$, or together terminated by regular terminal rules of the form $A \rightarrow a$, (A, B are nonterminals and a is a terminal symbol), thus generating the columns of the picture array. The words in the first phase form a context-sensitive language. An extension, called TCSML of the family CSML was introduced in [51] by having in a CSMG, an additional feature of tables of nonterminal rules or tables of terminal rules in the second phase. It was shown in [51] that $CSML \subset TCSML$. We show here that there are picture array languages that cannot be in TCSML but can be generated by APRPIS with two membranes.

Theorem 3. $AP_2(RPIS) - TCSML \neq \emptyset$

Proof. Consider the picture array language L_3 consisting of $(2m + 1) \times (2m + 1)$ arrays, ($m \geq 1$), such that an array in L_3 is of the form $(M_1 \circ M_2 \circ M_3) \diamond M_4 \diamond (M_3 \circ M_2 \circ M_1)$ where M_1, M_3 are $m \times m$ arrays over $\{a\}, \{b\}$ respectively, M_2 is an $m \times 1$ array over $\{e\}$ and M_4 is an $1 \times (2m + 1)$ array over $\{e\}$. A member of L_3 is shown in Figure 4. The language L_3 cannot belong to the family TCSML. In fact in a grammar generating a picture array in a language in TCSML, the number of rows above and below a distinct row of a generated array, cannot be maintained to be equal as the application of the tables of rules cannot be controlled.

The language L_3 is generated by the APRPIS Π_3 with two membranes where

$$\Pi_3 = (\{a, b, e\}, [1 \ 2]_2, F_1, F_2, R_1, R_2, 2)$$

with $F_1 = \left\{ \begin{smallmatrix} a & b \\ b & a \end{smallmatrix} \right\}$, $F_2 = \emptyset$. R_1 consists of the column insertion tables (c_1, in) , (c_2, in) and R_2 consists of the row insertion tables (r_3, out) , $(r_4, here)$ where

$$c_1 = \{(a, ab, b), (b, ba, a)\}, c_2 = \{(a, e, b), (b, e, a)\},$$

$$r_3 = \{(a, {}^t ab, b), (b, {}^t ba, a)\}, r_4 = \{(a, e, b), (b, e, a), (e, e, e)\}.$$

Starting with the axiom picture array in region 1, a computation in Π_3 can start in this region with the applicable tables of rules. If the rules of column insertion table c_1 are applied then ab will be inserted in every row between the two symbols a and b in two adjacent columns while ba will be inserted in every row between the two symbols b and a in the same two adjacent columns, thus adding two columns to the picture array. The array generated is sent to region 2. If in this region the rules of row insertion table r_3 is applied then ab will be inserted in all columns between the symbols a and b in two adjacent rows while ba will be inserted between the symbols b and a in the same two adjacent rows, thus adding two rows to the picture array and the array is sent back to region 1. The process can

repeat. If at any time, the rules of the column insertion table c_2 are applied in region 1, then a column of e 's will be inserted between the two adjacent columns ${}^t a \cdots ab \cdots b$ and ${}^t b \cdots ba \cdots a$ and the array is sent to region 2. A correct sequence of application of the tables of rules is to apply now in region 2, the rules of the row table r_4 , thus generating a picture array of the form shown in Figure 4. Region 2 is the output region and arrays collected here form the picture array language generated which is L_3 . \square

$$\begin{array}{cccccccc} a & \cdots & a & e & b & \cdots & b \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ a & \cdots & a & e & b & \cdots & b \\ e & \cdots & e & e & e & \cdots & e \\ b & \cdots & b & e & a & \cdots & a \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ b & \cdots & b & e & a & \cdots & a \end{array}$$

Figure 4. A picture array generated by Π_3 .

Head [52] introduced splicing systems while proposing a theoretical model of DNA recombination. Berstel et al. [53] introduced a variant of splicing system, known as flat splicing system and noted ([53] P4) that the flat splicing system involving the operation of flat splicing on words and the insertion system [40] involving the operation of insertion on words, have similarity but the systems are incomparable. Extending the operation of flat splicing to picture arrays, a picture array generating model, called array flat splicing system (AFS), was introduced and investigated in [54]. Here we show that picture array languages that cannot be generated by any AFS can be generated by APRPIS with two membranes. The family of picture array languages generated by AFS is denoted by $L(\text{AFS})$.

Theorem 4. $AP_2(\text{RPIS}) - L(\text{AFS}) \neq \phi$

Proof. Consider the picture array language L_4 consisting of $m \times (n + 2)$ arrays, ($m \geq 2, n \geq 1$), such that an array in L_4 is of the form $M_1 \circ M_2 \circ M_3$ where M_1, M_3 are $m \times 1$ arrays over $\{x\}, \{y\}$ respectively and M_2 is an $m \times n$ array over $\{a\}$. A member of L_4 is shown in Figure 5. The language L_4 does not belong to the family $L(\text{AFS})$ [54].

The language L_4 is generated by the APRPIS Π_4 with two membranes where

$$\Pi_4 = (\{x, y, a\}, [1 \ 2]_2, F_1, F_2, R_1, R_2, 2)$$

with $F_1 = \left\{ \begin{smallmatrix} x & a & y \\ x & a & y \end{smallmatrix} \right\}$. $F_2 = \phi$. R_1 consists of the column insertion tables. (c_1, here) , (c_2, in) and a row insertion table (r, here) where

$$c_1 = \{(x, a, a)\}, c_2 = \{(a, a, y)\}, r = \{(x, x, \lambda), (y, y, \lambda), (a, a, \lambda)\}.$$

We note that the rule of the column insertion table c_1 inserts a column of a 's but the array remains in the region 1 itself. The rule of the column insertion table c_2 has the same effect but the array is sent to the output region 2. The rule of the row insertion table r inserts a row of the form $xa \cdots ay$ while an array is generated. \square

$$\begin{array}{cccccc} x & a & a & a & a & y \\ x & a & a & a & a & y \\ x & a & a & a & a & y \\ x & a & a & a & a & y \end{array}$$

Figure 5. A picture array generated by Π_4 .

A class of array grammars called parallel contextual array grammars was considered in [55]. Several variants were introduced and their properties established. External parallel contextual array grammar is one such variant generating rectangular picture arrays and the family of picture array languages generated by this kind of array grammars is denoted by EPCA. Here we show that there are picture array languages not in EPCA, which can be generated by (*APRPIS*).

Theorem 5. $AP_2(RPIS) - EPCA \neq \emptyset$

Proof. Consider the picture array language $L_5 = L'_5 \cup L''_5$. L'_5 consists of $3 \times (2n + 1)$ arrays, ($n \geq 0$), such that an array M' in L'_5 is over $\{a\}$ and is of the form as shown in Figure 6. L''_5 consists of $3 \times (2n + 3)$ arrays, ($n \geq 0$), such that an array M'' in L''_5 is over $\{a, b\}$ and is of the form as shown in

Figure 7. The first and last columns of M'' are respectively b and a while all other symbols are a .

The language L_5 has been proved to be not in the family EPCA ([55] Lemma 4.5).

But the language L_5 is generated by the *APRPIS* Π_5 with two membranes where

$$\Pi_5 = (\{a, b\}, [1 [2]_2]_1, F_1, F_2, R_1, R_2, 2)$$

with $F_1 = F_2 = \{M_1 = \begin{array}{ccc} a & a & a \\ a & a & a \\ a & a & a \end{array}, M_2 = \begin{array}{ccc} a & a & b \\ b & a & a \\ a & a & b \end{array}\}$. R_1 consists of the column insertion tables. $(c_1, here), (c_2, in)$ where $c_1 = \{(a, aa, a), (a, aa, b)\}$, $c_2 = \{(a, aa, \lambda)\}$. $R_2 = \emptyset$.

Since region 2 is the output membrane, the axiom arrays in this region will be collected in the language generated. There are no rules in region 2. So in a computation in Π_5 only the rules in region 1 can be applied to the arrays in region 1. Initially, the rules of the table c_1 as well as the rule of the table c_2 are applicable and any of these should be applied to both the arrays M_1 and M_2 . If the rules of the table c_1 are applied to both M_1 and M_2 , then two columns of a 's are inserted yielding arrays

$$\begin{array}{cccccc} a & a & a & a & b & a & a & a & a & a \\ b & a & a & a & a & , & a & a & a & a \\ a & a & a & a & b & a & a & a & a & a \end{array}$$

but the arrays remain in the same region 1 as the column table c_1 has target *here* associated with it. The process can repeat. If the rule of the table c_2 is applied to any of the arrays generated in an earlier step (including the axiom arrays), then again two columns of a 's are inserted into the array but the generated array is sent to output region 2. Since there are no rules in region 2, these arrays cannot evolve further and are collected in the language generated by Π_5 . The language generated is L_5 . \square

$$\begin{array}{ccccc} a & a & a & a & a \\ a & a & a & a & a \\ a & a & a & a & a \end{array}$$

Figure 6. A picture array in L'_5 .

$$\begin{array}{cccccc} a & a & a & a & a & a & b \\ b & a & a & a & a & a & a \\ a & a & a & a & a & a & b \end{array}$$

Figure 7. A picture array in L''_5 .

7. An Application of the Model to “Kolam” Pattern Generation

A “kolam” pattern [43] is a visually pleasing floor design, more common in South India, drawn with curly lines around points arranged in a particular pattern, resulting in the intended “kolam” drawing. In [43] “kolam” patterns were considered as picture arrays in the two-dimensional plane with the labels of the cells of the picture array representing primitive “kolam” patterns [56]. In fact array grammars developed in [43] were used to generate the picture arrays and the labels were replaced by the corresponding primitive patterns, thus yielding the kolam pattern composed of these primitive patterns. Here we illustrate by constructing a *APRPIS* generating picture arrays representing a particular set of “kolam patterns”, one member of which is shown in Figure 8. The labels of the arrays stand for the primitive kolam patterns defining the “kolam”. An array representing this kind of a kolam is shown in Figure 9.

Let L_k be the picture language consisting of $(m + 2) \times (2n + 3)$, $m \geq 1, n \geq 1$ picture arrays p such that $p(i, 1) = p(i, n + 2) = p(i, 2n + 3) = b$, for $i \in \{1, m + 2\}$, $p(1, j) = d_1$ for $2 \leq j \leq n + 1; n + 3 \leq j \leq 2n + 2$, $p(n + 2, j) = d_2$ for $2 \leq j \leq n + 1; n + 3 \leq j \leq 2n + 2$, $p(i, 1) = d_3$, for $2 \leq i \leq n + 1$, $p(i, 2n + 3) = d_4$, for $2 \leq j \leq n + 1$, $p(i, n + 2) = e$, for $2 \leq i \leq n + 1$ and all other $p(i, j) = d$. Here d_1, d_2, d_3, d_4, d, e stand for the “kolam” primitive patterns as in Figure 10 and b stands for blank.

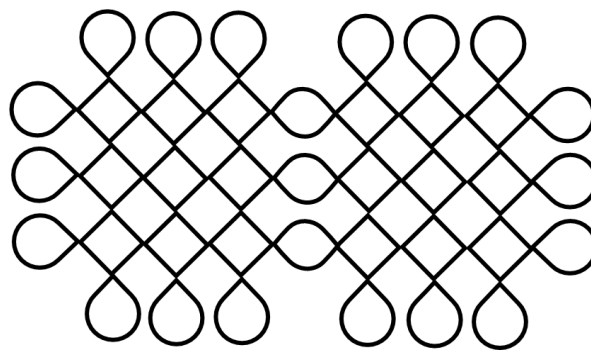


Figure 8. A “Kolam” pattern.

b	d_1	d_1	d_1	b	d_1	d_1	d_1	b
d_3	d	d	d	e	d	d	d	d_4
d_3	d	d	d	e	d	d	d	d_4
d_3	d	d	d	e	d	d	d	d_4
b	d_2	d_2	d_2	b	d_2	d_2	d_2	b

Figure 9. Array representing the “kolam” in Figure 8.

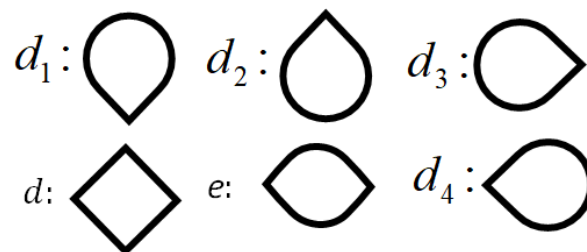


Figure 10. A set of primitive “Kolam” patterns.

The picture language L_k is generated by the *APRPIS* with alphabet. $\{d_1, d_2, d_3, d_4, d, e, b\}$,
 membrane structure $[1 [2 [3]3]2]1$, axiom array $b \ d_1 \ b \ d_1 \ b$
 $d_3 \ d \ e \ d \ d_4$ in region 1 and no axiom array
 $b \ d_2 \ b \ d_2 \ b$
 in regions 2 and 3, with 3 as the output region. The sets R_1, R_2, R_3 are the sets of tables of insertion

rules in regions 1,2 and 3 respectively, where R_1 contains a column insertion table (c_1, in) and a row insertion table ($r_1, here$), R_2 contains two column insertion tables (c_2, out) and (c_3, in). Here

$$c_1 = \{(d_1, d_1, b), (d, d, e), (d_2, d_2, b)\}, c_2 = \{(b, d_1, d_1), (e, d, d), (b, d_2, d_2)\}$$

$$c_3 = \{(d_1, d_1, b), (d, d, d_4), (d_2, d_2, b)\},$$

$$r_1 = \{(b, d_3, d_3), (d_1, d, d), (b, e, e), (b, d_4, d_4)\}.$$

In a computation, the rules of table c_1 insert a column of the form ${}^t d_1 d \cdots d d_2$ to the left of the middle column of e 's while the rules of table c_2 insert a similar column to the right of the middle column of e 's. The generated array is sent to region 2 from region 1 on applying c_1 while it is sent from region 2 to region 1 on applying c_2 . The process can repeat adding equal number of columns to the left and right of the middle column. The application of the rules of the table r_1 inserts below the first row, a row of symbols $d_3 d \cdots d e d \cdots d d_4$ as many times as needed and the generated array remains in region 1. When the rules of the table c_3 are applied in region 2, it has the same effect as applying c_2 but the insertion is to the left of the last column and the array is sent to the output region 3, with the computation halting. The generated arrays are the picture arrays of L_k and no other array is generated. On replacing the labels of the squares in a generated picture array by the corresponding primitive kolam patterns shown in Figure 10, we obtain the “kolam” itself. One such generated “kolam” is shown in Figure 8. It is interesting to note that the “kolam” pattern in Figure 8 has a mathematically pleasing feature of symmetrical subpatterns on the left and right of the middle line made of the primitive pattern e shown in Figure 10.

8. Concluding Remarks

In the framework of membrane computing, a new picture array generating model, called array P System with restricted picture insertion rules (*APRIS*) has been introduced. The generative power of this model is exhibited by comparing it with many standard families of picture languages. The main theoretical contribution is that the new model *APRIS* can generate picture array languages which cannot be generated by the other picture array grammar models considered here. It will be of interest to compare with other families of picture languages not considered here such as the family of picture languages generated by internal parallel contextual array grammars [55]. As an application of this new theoretical model, certain floor-designs, called “kolam” patterns are generated, thereby showing that the model can handle the problem of generation of patterns encoded as picture arrays over a finite set of symbols.

Author Contributions: Conceptualization of this research was done by K.G.S.; Investigation of theoretical results established and applications were done by G.S. and N.G.D.; Writing and preparation of the draft version were done by K.G.S. and G.S.; Writing and editing and providing suggestions for improving the contents were done by G.Z. and A.K.N. Final submission version was prepared with the involvement of all the authors. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Acknowledgments: The authors thank the reviewers for their useful comments which enabled them to improve the presentation of the paper. This work was supported by the National Natural Science Foundation of China (61972324, 61672437, 61702428), by Beijing Advanced Innovation Center for Intelligent Robots and Systems (2019IRS14), New Generation Artificial Intelligence Science and Technology Major Project of Sichuan Province (2018GZDZX0043) and Artificial Intelligence Key Laboratory of Sichuan Province (2019RYJ06).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Păun, G. *Membrane Computing: An Introduction*; Springer: Berlin/Heidelberg, Germany, 2000.
2. Păun, G.; Rozenberg, G.; Salomaa, A. *The Oxford Handbook of Membrane Computing*; Oxford University Press Inc.: New York, NY, USA, 2010.
3. Pan, L.; Păun, G.; Zhang, G. Foreword: Starting JMC. *J. Membr. Comput.* **2019**, *1*, 1–2. [\[CrossRef\]](#)
4. Păun, G. Computing with membranes. *J. Comp. System Sci.* **2000**, *61*, 108–143. [\[CrossRef\]](#)
5. Zhang, G.; Pérez-Jiménez, M.J.; Gheorghe, M. Real-life Applications with Membrane Computing. In *Emergence, Complexity and Computation Series*; Springer: Berlin/Heidelberg, Germany, 2017.
6. Wang, J.; Shi, P.; Peng, H. Membrane computing model for IIR filter design. *Inf. Sci.* **2016**, *329*, 164–176. [\[CrossRef\]](#)
7. Zhang, G.; Rong, H.; Neri, F.; Pérez-Jiménez, M.J. An optimization spiking neural P system for approximately solving combinatorial optimization problems. *Int. J. Neural Syst.* **2014**, *24*, 1440006. [\[CrossRef\]](#)
8. Zhang, G.; Gheorghe, M.; Pan, L.; Pérez-Jiménez, M.J. Evolutionary membrane computing: A comprehensive survey and new results. *Inf. Sci.* **2014**, *279*, 528–551. [\[CrossRef\]](#)
9. Andreu-Guzmán, J.A.; Valencia-Cabrera, L. A novel solution for GCP based on an OLMS membrane algorithm with dynamic operators. *J. Membr. Comput.* **2019**, *1*, 1–13. [\[CrossRef\]](#)
10. Wang, X.; Zhang, G.; Neri, F.; Jiang, T.; Zhao, J.; Gheorghe, M.; Ipate, F.; Lefticaru, L. Design and implementation of membrane controllers for trajectory tracking of nonholonomic wheeled mobile robots. *Integr. Comput. Aided Eng.* **2016**, *23*, 15–30. [\[CrossRef\]](#)
11. Perez-Hurtado, I.; Martinez-del-Amor, M.A.; Zhang, G.; Neri, F.; Pérez-Jiménez, M.J. A membrane parallel rapidly-exploring random tree algorithm for robotic motion planning. *Integr. Comput. Aided Eng.* **2020**, *27*, 121–138. [\[CrossRef\]](#)
12. Buiu, C.; George, A. Membrane computing models and robot controller design, current results and challenges. *J. Membr. Comput.* **2019**, *1*, 262–269. [\[CrossRef\]](#)
13. Peng, H.; Wang, J.; Ming, J.; Shi, P.; Pérez-Jiménez, M.J.; Yu, W.; Tao, C. Fault diagnosis of power systems using intuitionistic fuzzy spiking neural P systems. *IEEE Trans. Smart Grid* **2018**, *9*, 4777–4784. [\[CrossRef\]](#)
14. Wang, T.; Zhang, G.; Zhao, J.; He, Z.; Wang, J.; Pérez-Jiménez, M.J. Fault diagnosis of electric power systems based on fuzzy reasoning spiking neural P systems. *IEEE Trans. Power Syst.* **2015**, *30*, 1182–1194. [\[CrossRef\]](#)
15. Rong, H.; Yi, K.; Zhang, G.; Dong, J.; Paul, P.; Huang, Z. Automatic implementation of fuzzy reasoning spiking neural P systems for diagnosing faults in complex power systems. *Complexity* **2019**, *2019*, 2635714. [\[CrossRef\]](#)
16. Sánchez-Karhunen, E.; Valencia-Cabrera, L. Modelling complex market interactions using PDP systems. *J. Membr. Comput.* **2019**, *1*, 40–51. [\[CrossRef\]](#)
17. Díaz-Pernil, D.; Gutiérrez-Naranjo, M.A.; Peng, H. Membrane computing and image processing: A short survey. *J. Membr. Comput.* **2019**, *1*, 58–73. [\[CrossRef\]](#)
18. Peng, H.; Wang, J.; Shi, P.; Pérez-Jiménez, M.J.; Riscos-Núñez, A. An extended membrane system with active membrane to solve automatic fuzzy clustering problems. *Int. J. Neural Syst.* **2016**, *26*, 1–17. [\[CrossRef\]](#)
19. Orellana-Martín, D.; Valencia-Cabrera, L.; Riscos-Núñez, A.; Pérez-Jiménez, M.J. P systems with proteins: A new frontier when membrane division disappears. *J. Membr. Comput.* **2019**, *1*, 29–39. [\[CrossRef\]](#)
20. Jiang, Y.; Su, Y.; Luo, F. An improved universal spiking neural P system with generalized use of rules. *J. Membr. Comput.* **2019**, *1*, 270–278. [\[CrossRef\]](#)
21. Freund, R. How derivation modes and halting conditions may influence the computational power of P systems. *J. Membr. Comput.* **2020**, *2*, 14–25. [\[CrossRef\]](#)
22. Leporati, A.; Manzoni, L.; Mauri, G.; Porreca, A.E.; Zandron, C. A Turing machine simulation by P systems without charges. *J. Membr. Comput.* **2020**, *2*, 71–79. [\[CrossRef\]](#)
23. Orellana-Martín, D.; Valencia-Cabrera, L.; Riscos-Núñez, A.; Pérez-Jiménez, M.J. Minimal cooperation as a way to achieve the efficiency in cell-like membrane systems. *J. Membr. Comput.* **2019**, *1*, 85–92. [\[CrossRef\]](#)
24. Leporati, A.; Manzoni, L.; Mauri, G.; Porreca, A.E.; Zandron, C. Characterizing PSPACE with shallow non-confluent P systems. *J. Membr. Comput.* **2019**, *1*, 75–84. [\[CrossRef\]](#)
25. Cooper, J.; Nicolescu, R. Alternative representations of P systems solutions to the graph colouring problem. *J. Membr. Comput.* **2019**, *1*, 112–126. [\[CrossRef\]](#)

26. Sosík, P. *P* systems attacking hard problems beyond NP: a survey. *J. Membr. Comput.* **2019**, *1*, 198–208. [[CrossRef](#)]
27. Calude, C.S.; Dinneen, M.J.; Hua, R. Quantum solutions for densest *k*-subgraph problems. *J. Membr. Comput.* **2020**, *2*, 26–41. [[CrossRef](#)]
28. Buño, K.; Adorna, H. Distributed computation of a *k P* systems with active membranes for SAT using clause completion. *J. Membr. Comput.* **2020**, *2*, 108–120. [[CrossRef](#)]
29. Chen, H.; Freund, R.; Ionescu, M.; Păun, G.; Pérez-Jiménez, M.J. On string languages generated by spiking neural *P* systems. *Fundam. Inform.* **2007**, *75*, 141–162.
30. Ferretti, C.; Mauri, G.; Păun, G.; Zandron, C. On three variants of rewriting *P* systems. *Theor. Comp. Sci.* **2003**, *301*, 201–215. [[CrossRef](#)]
31. Krithivasan, K.; Metta, V.P.; Garg, D. On string languages generated by spiking neural *P* systems with anti-spikes. *Int. J. Found. Comput. Sci.* **2011**, *22*, 15–27. [[CrossRef](#)]
32. Cruz, R.T.A.D.L.; Cabarle, F.G.; Adorna, H.N. Generating context-free languages using spiking neural *P* systems with structural plasticity. *J. Membr. Comput.* **2019**, *1*, 161–177. [[CrossRef](#)]
33. Ceterchi, R.; Mutyam, M.; Păun, G.; Subramanian, K.G. Array-rewriting *P* systems. *Nat. Comput.* **2003**, *2*, 229–249. [[CrossRef](#)]
34. Subramanian, K.G. *P* systems and picture languages. In *Lecture Notes in Computer Science*; Springer: Berlin/Heidelberg, Germany, 2007; Volume 4664, pp. 99–109.
35. Subramanian, K.G.; Sriram, S.; Song, B.; Pan, L. An Overview of 2D Picture Array Generating Models Based on Membrane Computing. In *Reversibility and Universality. Emergence, Complexity and Computation*; Adamatzky, A., Ed.; Springer: Berlin/Heidelberg, Germany, 2018; Volume 30, pp. 333–356.
36. Fujioka, K. Morphic characterizations with insertion systems controlled by a context of length one. *Theoret. Comput. Sci.* **2013**, *469*, 69–76. [[CrossRef](#)]
37. Haussler, D. Insertion languages. *Inf. Sci.* **1983**, *31*, 77–89. [[CrossRef](#)]
38. Margenstern, M.; Păun, G.; Rogozhin, Y.; Verlan, S. Context-free insertion-deletion systems. *Theor. Comp. Sci.* **2005**, *330*, 339–348. [[CrossRef](#)]
39. Kari, L.; Paun, G.; Thierrin, G.; Yu, S. At the crossroads of DNA computing and formal languages: Characterizing recursively enumerable languages using insertion-deletion systems. *DIMACS Ser. Discret. Math. Theor. Comput. Sci.* **1999**, *48*, 329–338.
40. Păun, G.; Rozenberg, G.; Salomaa, A. *DNA Computing: New Computing Paradigms*; Springer: New York, NY, USA, 1998.
41. Fujioka K. A Two-Dimensional Extension of Insertion Systems. In *Theory and Practice of Natural Computing (TPNC 2014)*; Dediu, A.H., Lozano, M., Martín-Vide, C., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2014; Volume 8890, pp. 181–192.
42. Subramanian, K.G.; Ali, R.M.; Geethalakshmi, M.; Nagar, A.K. Pure 2D picture grammars and languages. *Discret. Appl. Math.* **2009**, *157*, 3401–3411. [[CrossRef](#)]
43. Siromoney, G.; Siromoney, R.; Krithivasan, K. Array grammars and Kolam *Comput. Graph. Image Proc.* **1974**, *3*, 63–82. [[CrossRef](#)]
44. Samdanielthompson, G.; Nagar, A.K.; David, N.G.; Zhang, G.; Subramanian, K.G. Insertion Based Picture Array *P* systems. In *Proceedings of the ACMC 2018, CDMTCS Research Report Series (CDMTCS-530)*, Auckland, New Zealand, 10–14 December 2018; Dinneen, M.J., Nicolescu, R., Eds.; University of Auckland: Auckland, New Zealand, 2018.
45. Rozenberg, G.; Salomaa, A. *Handbook of Formal Languages*; Springer: Berlin/Heidelberg, Germany, 1997.
46. Giammarresi, D.; Restivo, A. Two-dimensional languages. In *Handbook of Formal Languages*; Rozenberg, G., Salomaa, G., Eds.; Springer: Berlin/Heidelberg, Germany, 1997; pp. 215–267.
47. Morita, K. Two-dimensional languages. In *Formal Languages and Applications. Series in Fuzziness and Soft Computing*; Martín-Vide, C., Mitrana, V., Păun, G., Eds.; Springer: Berlin/Heidelberg, Germany, 2004; Volume 148, pp. 426–437.
48. Krivka, Z.; Martín-Vide, C.; Meduna, A.; Subramanian, K.G. A Variant of Pure Two-Dimensional Context-Free Grammars Generating Picture Languages. In *IWCIA 2014, Lecture Notes in Computer Science*; Springer: Berlin/Heidelberg, Germany, 2014; Volume 8466, pp. 123–133.

49. Subramanian, K.G.; Geethalakshmi, M.; David, N.G.; Nagar, A.K. Picture Array Generation Using Pure 2D Context-Free Grammar Rules. In *IWCIA 2015 Lecture Notes in Computer Science*; Springer: Berlin/Heidelberg, Germany, 2015; Volume 4664, pp. 187–201.
50. Siromoney, G.; Siromoney, R.; Krithivasan, K. Abstract families of matrices and picture languages. *Comput. Graph. Image Proc.* **1972**, *1*, 284–307. [[CrossRef](#)]
51. Siromoney, R.; Subramanian, K.G.; Rangarajan, K. Parallel/sequential rectangular arrays with tables. *Int. J. Comput. Math.* **1977**, *6*, 143–158. [[CrossRef](#)]
52. Head, T. Formal language theory and DNA: An analysis of the generative capacity of specific recombinant behaviours. *Bull. Math. Biol.* **1987**, *49*, 735–759. [[CrossRef](#)]
53. Berstel, J.; Boasson, L.; Fagnot, I. Splicing systems and the Chomsky hierarchy. *Theor. Comput. Sci.* **2012**, *436*, 2–22. [[CrossRef](#)]
54. Pan, L.; Nagar, A.K.; Subramanian, K.G.; Song, B. Picture Array Generation Using Flat Splicing Operation. *J. Comput. Theoret. Nanosci.* **2016**, *13*, 3568–3577. [[CrossRef](#)]
55. Subramanian, K.G.; Van, D.L.; Helen Chandra, P.; Quyen, N.D. Array Grammars with Contextual Operations. *Fundam. Inform.* **2008**, *83*, 411–428.
56. Nagata, S.; Robinson, T. Digitalization of Kolam Patterns and Tactile Kolam Tools. In *Formal Models, Languages and Applications*; Subramanian, K.G., Rangarajan, K., Madhavan, M., Eds.; World Scientific Publishing: Singapore, 2007; pp. 354–363.

Publisher’s Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).